УДК 004.4'423

Ю.А. Зорин

Интерпретатор языка построения генераторов тестовых заданий на основе деревьев И/ИЛИ

Предложен язык разработки генераторов тестовых заданий GILT, основанный на методах генерации комбинаторных множеств деревьев И/ИЛИ. Рассматривается синтаксический анализ языка и его реализация на функциональном языке F#, а также пример построения алгоритма генерации тестового задания на языке GILT.

Ключевые слова: генерация тестовых заданий, функциональный язык, дерево И/ИЛИ.

Автоматизированные системы генерации тестовых заданий являются важной частью современных систем дистанционного обучения [1]. Эти системы, как правило, обладают некоторым языком описания генераторов тестовых заданий и интерпретатором, позволяющим получить конкретный вариант задания в виде некоторого файла в заданном формате. В свою очередь, реализация алгоритмов генерации в подобных системах основана на применении различных языков программирования:

- общего назначения (C++, C, Pascal и др.) [2];
- специального назначения (Possum, «Фея») [1, 3];
- системы компьютерной алгебры (Mathematica, Maxima, MathCad).

Существует также ряд библиотек функций и классов, реализующих алгоритмы генерации для языков программирования общего назначения. К подобным относится интерпретатор языка GIL (Generation and Identification Language). Данный язык предназначен для построения генераторов комбинаторных множеств на основе деревьев И/ИЛИ [4]. Язык предусматривает различные операции над деревьями И/ИЛИ и также применим для описания генераторов тестовых заданий в виде узлов дерева И/ИЛИ, с определенными ограничениями:

- отсутствие операторов выбора варианта дерева по условию;
- отсутствие именованных узлов для идентификации варианта.

Подобные ограничения не позволяют описать генератор вопроса с различными решениями в зависимости от варианта дерева И/ИЛИ, а также корректировать условие задания в зависимости от ранее выбранных параметров. В свою очередь, метод описания генераторов тестовых заданий в виде деревьев И/ИЛИ позволит производить идентификацию генератора задания, а также его вариантов, рассчитывать мощность генератора, что позволит получать вариант задания по его номеру. Последние утверждения показывают необходимость в развитии языка GIL и создании на его основе синтаксиса языка и его интерпретатора для построения генераторов тестовых заданий на основе деревьев И/ИЛИ (GILT – Generation and Identification Language Tests).

В данной статье описывается реализация интерпретатора языка GILT и его возможности для генерации тестовых заданий.

Язык генерации тестовых заданий GILT. Язык GILT является развитием функционального языка GIL. Язык позволяет описывать узлы дерева И/ИЛИ в виде скобочной нотации. Для записи узла И предлагается использовать круглые скобки, а для узла ИЛИ – фигурные, именованные узлы содержат название перед скобками (рис. 1).

Введем понятие «условно-именованный узел» (обозначается символом решетки – «#», устанавливаемый перед именным узлом), а также «условный узел». Условно-именованные узлы участвуют при инициализации дерева и могут переопределяться в зависимости от выбранного варианта. Условный узел является узлом И, состоящим из трех сыновей: условие, первый сын при условии True, второй – False. Узлы дерева могут быть нескольких типов:

- математическое выражение;
- строка;
- дерево;
- условное выражение (в условных узлах).

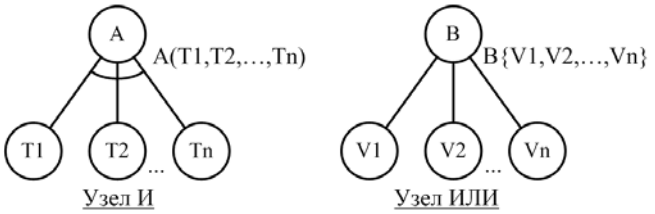

Рис. 1. Скобочная нотация записи узлов И/ИЛИ

Любой алгоритм генерации на языке GILT представляется в виде дерева И/ИЛИ. Результатом работы является получение варианта при левостороннем обходе дерева и выборе варианта (случайным образом либо по номеру) каждого из узлов ИЛИ (рис. 2).

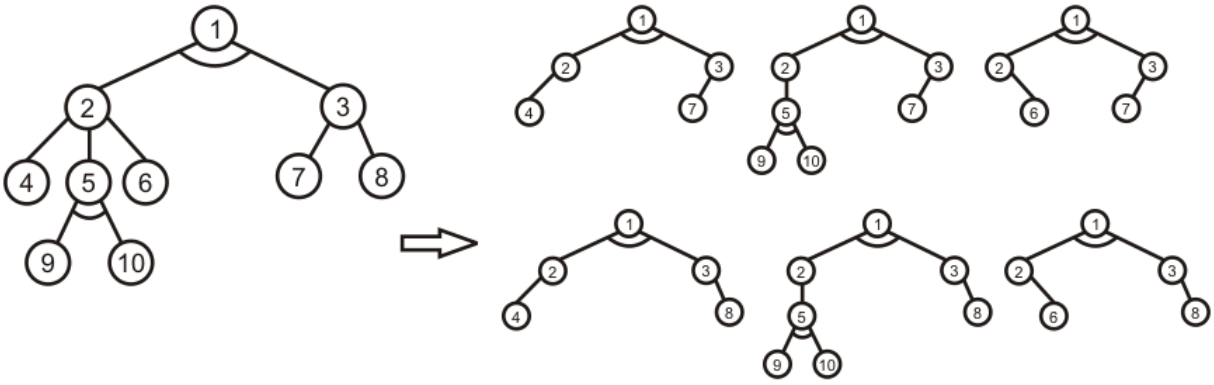

Рис. 2. Дерево И/ИЛИ и все его варианты

Пример построения генератора тестовых заданий на языке GILT. Рассмотрим пример построения генератора задачи по теории вероятности и его описание на языке GILT.

В ящике имеется 10 белых и 15 черных шаров. Вынимается 4 шара. Какова вероятность того, что все вынутые шары будут одного цвета?

Для описания алгоритма генерации данной задачи представим текст как совокупность строковых символов. Данное действие позволит получать различные варианты представления данной задачи за счет изменения заранее определенных групп символов (слов, чисел и т.д.). Весь текст условия задачи разделим на три фрагмента {A, B, C}, тогда узел Q будет И-узлом, содержащим сыновей A, B, C. Каждый из фрагментов разделяется на фиксированные и переменные части. Например, узел B разбивается на три фиксированных фрагмента {T4, T8, T12} и два переменных {V2, V3}, которые имеют по три варианта реализации. Ниже в табл. 1 перечислены значения узлов $\{TN\}_{N=1}^{15}$ [5].

Таблица 1

Значения узлов

T1 – B	T9 – 8
T2 – ящике	T10 – 10
T3 – коробке	T11 – 7
T4 – имеется	T12 – черных шаров
T5 – 10	T13 – вынимается
T6 – 15	T14 – 2 шара. Какова вероятность того, что вынутые шары будут разных цветов.
T7 – 20	T15 – 4 шара. Какова вероятность того, что все вынутые шары будут одинакового цвета
T8 – белых	

Для наглядного отображения всех возможных вариантов условий данной задачи, построим дерево И/ИЛИ (рис. 3).

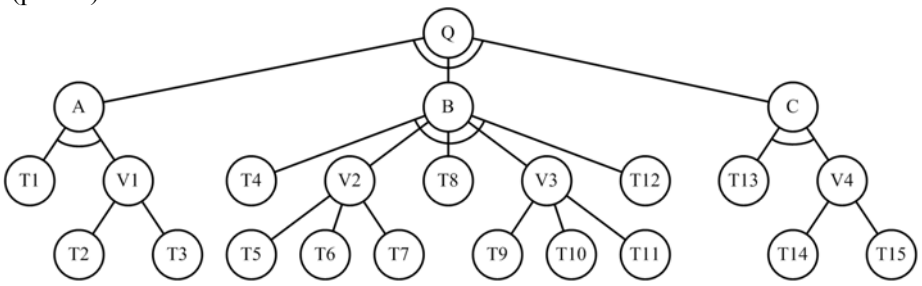

Рис. 3. Схематичное представление задачи в виде дерева И/ИЛИ

Используя правила записи узлов [4]: узлы И – обозначаются в круглых скобках, узлы ИЛИ – в фигурных скобках, именованные узлы содержат название перед скобками, запишем дерево (см. рис. 2) на языке GILT:

```
Q(
A(T1, V1{T2, T3}),
B(T4, V2{T5, T6, T7}, T8, V3{T9, T10, T11}, T12),
C(T13, V4{T14, T15})
)
```

Условие задачи может иметь некоторую другую интерпретацию рассмотренного тестового задания, в котором меняется условия задания в зависимости от варианта одного из узлов:

В ящике имеется 10 белых и 15 черных шаров. Из ящика вынимаются 4 шара. Какова вероятность того, что все вынутые шары будут белого цвета?

При попытке вставки в изменяемый узел ИЛИ слова «белых» с вариантами «белых», «красных», «синих» возможен следующий исход задания:

В ящике имеется 10 красных и 15 черных шаров. Из ящика вынимается 4 шара. Какова вероятность того, что все вынутые шары будут белого цвета?

Как видно, в тексте вопроса осталось слово «белого», что делает задание некорректным. В предотвращении подобной ситуации необходимо использовать условно-именованные узлы, заменив узел T8 на узел ИЛИ и добавив узел T16 в переменный V4 следующим образом:

```
Q(
A(T1, V1{T2, T3}),
B(T4, V2{T5, T6, T7}, V5{(T17, #T16(“белого”)), (T18,
#T16(“красного”)), (T19, #T16(“синего”))}, V3{T9, T10, T11}, T12),
C(T13, V4{T14, T15, (T20, T16, T21)})
)
```

В табл. 2 приведены значения узлов $\{TN\}_{N=17}^{21}$.

Таблица 2

Значения узлов ИЛИ

T17 – белых и	T20 – 6 шаров. Какова вероятность того, что все вынутые шары будут
T18 – красных и	T21 – цвета
T19 – синих и	

Узел T16 будет инициализироваться в момент генерации того или иного варианта переменного узла V5, но его значение будет использовано, например, для вывода на экран, только в узле V4. Особенность условно-именных узлов, инициализирующихся только в момент выбора того или иного варианта дерева И/ИЛИ, позволит описывать ответы и решения для той или иной задачи.

Разработка интерпретатора GILT. Для реализации программного интерпретатора синтаксической записи дерева И/ИЛИ необходимо учесть ряд требований, предъявляемых к языку GILT:

- учесть рекурсивную природу деревьев;
- реализацию алгоритмов генерации без решения инфраструктурных задач;
- доступ к генераторам случайных чисел;
- наличие математических выражений;
- наличие условного оператора;
- возможность идентификации объектов для их последующего использования.

Функциональная парадигма программирования характеризуется рекурсивными функциями, тем самым предоставляя более удобные возможности обработки древовидных структур. Язык F# предоставляет полный набор инструментов функционального программирования: алгебраические типы данных, функции высшего порядка, средства для композиции функций и неизменяемые структуры данных. Все функциональные возможности F# реализованы поверх общей системы типов .NET Framework [6].

Неотъемлемой частью реализации интерпретатора является лексический и синтаксический анализ введенных пользователем строк, позволяющий переводить скобочную запись деревьев И/ИЛИ в описанный тип данных дерева И/ИЛИ. Среди различных анализаторов кода внимания заслуживает программная библиотека «Yacc», в частности, из-за того, что в своем составе содержит сразу лекси-

ческий и синтаксический анализаторы и позволяет выдавать результат на языке F# (FYacc). Для описания лексического анализатора на языке F# приведена синтаксическая диаграмма дерева И/ИЛИ (рис. 4).

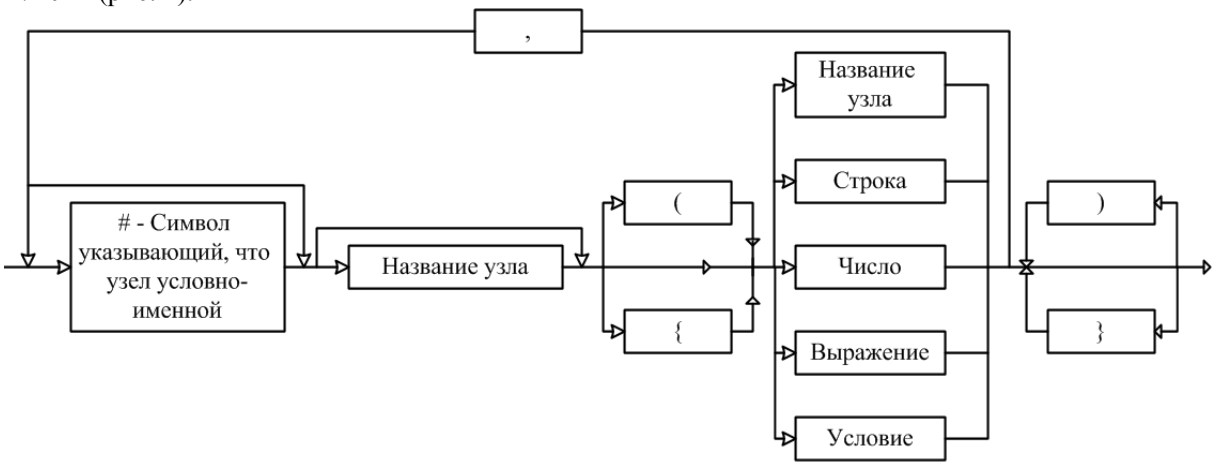

Рис. 4. Синтаксическая диаграмма языка GILT

Синтаксический анализатор на основе символьных конструкций, определенных за счет регулярных выражений, использует рекурсивный спуск при попытке приведения полученной строки к типу Tree, которое в свою очередь будет описано в размеченном объединении:

```
start:
  | Prog1           { Tree($1) }
Tree:
  | LOR Tree ROR         { Or ([] @ $2)}
  | LAND Tree RAND  { And ([] @ $2)}
  | NAMETREE Tree   { SetNameTree($1, $2) }
  | STR             { Str $1}
  | INT32           { Int $1}
  | FLOAT       { Float $1}
```

Размеченные объединения языка F# позволяют описать типы, учитывая рекурсивную природу деревьев И/ИЛИ, и, используя сопоставление с образцом, сразу приступать к применению различных операций над листьями и узлами дерева [6]. Размеченное объединение для дерева И/ИЛИ:

```
type AndOrTree =
//Узел может содержать один или несколько узлов И
| And of AndOrTree list
//Узел может содержать один или несколько узлов ИЛИ
| Or of AndOrTree list
//Листья дерева могут содержать целочисленные значения
| Int of int
//Листья дерева могут содержать строковые значения
| String of string
//Листья дерева могут содержать значения с плавающей точкой
| Float of float
//Узлы могут быть именованные
| SetNameTree of string * AndOrTree
//Узлы могут содержать имя узла, инициализованного ранее
| GetNameTree of string * AndOrTree
  // Обозначаем корневой узел
and Tree =
| Tree of AndOrTree
```

К основным характеристикам интерпретатора языка GILT относятся расширяемость и переносимость в различные проекты, разрабатываемые на платформе .NET.

Заключение. Реализация интерпретатора языка GILT даст возможность построения генераторов тестовых задач на основе деревьев И/ИЛИ, что позволит производить идентификацию генератора задания, а также его вариантов, рассчитывать мощность генератора и получать вариант задания

по его номеру. Скобочная нотация записи дерева И/ИЛИ позволит разрабатывать генераторы тестовых заданий на основе визуальных компонентов построения дерева в составе инструментальной системы разработки генераторов тестовых заданий и решить ряд проблем, связанных с отсутствием у преподавателей навыков программирования при разработке генераторов.

Литература

1. Посов И.А. Автоматизация процесса разработки и использования многовариантных учебных заданий: автореф. дис. … канд. техн. наук. – СПб.: СПбГУ, 2012. – 18 с.

2. Левинская М.А. Автоматизированная генерация заданий по математике для контроля знаний учащихся // Educational Technology & Society. – 2002. – Т. 5, вып. 4. – С. 214–221.

3. Кручинин В.В. Генераторы в компьютерных учебных программах. – Томск: Изд-во Том. ун-та, 2003. – 200 с.

4. Титков А.В. Система построения генераторов комбинаторных множеств на основе деревьев И/ИЛИ: автореф. дис. … канд. техн. наук / А.В. Титков. – Томск: ТУСУР, 2010. – 22 с.

5. Кручинин В.В. Методы генерации тестовых заданий по информатике // Информатика и образование (Москва). – 2005. – № 2. – С. 87–93.

6. Смит К. Программирование на F#: учеб. пособие. – М.: Символ-Плюс, 2011. – 448 с.

Зорин Юрий Алексеевич
Аспирант каф. промышленной электроники ТУСУРа
Тел.: 8(3822) 42-30-67
Эл. почта: yura@freebrains.ru

Zorin Yu.A.

The interpreter of programming language for design generators of tests based on AND/OR trees

The language of design generators of tests GILT, based on the methods of generating combinatorial sets of AND/OR trees. The article deals with parsing the language and its implementation in the functional language F#, as well as the example of the designed algorithm for generating the tasks in GILT.

Keywords: generator of tests, functional language, AND/OR tree.
